[illegible]

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/821,444	03/29/2001	Junji Sakai	P/2041-58	1745

7590 12/02/2004

Steven Dickson
1177 Avenue of the americas
New York, NY 10036-2714

EXAMINER
STEELMAN, MARY J

ART UNIT	PAPER NUMBER
2122	

DATE MAILED: 12/02/2004

Please find below and/or attached an Office communication concerning this application or proceeding.

PTO-90C (Rev. 10/03)

Office Action Summary	Application No.	Applicant(s)
	09/821,444	SAKAI, JUNJI
	Examiner	Art Unit
	Mary J. Steelman	2122

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If the period for reply specified above is less than thirty (30) days, a reply within the statutory minimum of thirty (30) days will be considered timely.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *24 August 2004*.

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-33* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-33* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☒ The drawing(s) filed on *24 August 2004* is/are: a)☐ accepted or b)☒ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☒ Information Disclosure Statement(s) (PTO-1449 or PTO/SB/08) Paper No(s)/Mail Date *29 March 2001*.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____ .
5) ☐ Notice of Informal Patent Application (PTO-152)
6) ☐ Other: _____.

DETAILED ACTION

1. This action is in response to Amendment and Remarks received 24 August 2004. Per Applicant's request, claims 2-10, 12-14, 16-27 and 29-33 have been amended. Claims 1-33 are pending.

2. The prior, non-final office action is hereby withdrawn and replaced with this current non-final office action.

Information Disclosure Statement

3. Regarding Applicant's remarks referencing form 1449. Examiner has initialed as considered the Abstract of Document Number 10078880, also noted as Document 3.

Regarding the other documents: Page 2, lines 3-10 identifies Documents 1, 2, and 3.
Document 1 is not discussed.
The relevance of Document 2 is discussed in detail in the Background of the Invention section of the specification beginning at page 2, line 11.
Document 4 is identified and discussed at page 7, line 22 through page 9, line 25.
As Documents 2 and 4 have been summarized with some detail, Examiner has initialed them as considered.

Document 5 is minimally addressed in a brief sentence at page 24, lines 17-21 and again at page 32, lines 24-26. Although an English version of the Aho text is available (Document 5), Examiner cannot verify that pages in a Japanese version of the Aho book relate to certain pages of an English version. See Specification, page 24, line 20 and page 32, line 25.

Therefore, Examiner disagrees with Applicant's statement that documents 1 and 5 were discussed in detail in the Background of the Invention section of the Specification, as noted on page 29, 2nd paragraph.

Drawings

4. Drawings received 18 August must be identified in the top margin as "Replacement Sheet".

INFORMATION ON HOW TO EFFECT DRAWING CHANGES

Replacement Drawing Sheets

Drawing changes must be made by presenting replacement figures which incorporate the desired changes and which comply with 37 CFR 1.84. An explanation of the changes made must be presented either in the drawing amendments, or remarks, section of the amendment. Any replacement drawing sheet must be identified in the top margin as "Replacement Sheet" (37 CFR 1.121(d)) and include all of the figures appearing on the immediate prior version of the sheet, even though only one figure may be amended. The figure or figure number of the amended drawing(s) must not be labeled as "amended." If the changes to the drawing figure(s) are not accepted by the examiner, applicant will be notified of any required corrective action in the next Office action. No further drawing submission will be required, unless applicant is notified.

Identifying indicia, if provided, should include the title of the invention, inventor's name, and application number, or docket number (if any) if an application number has not been assigned to the application. If this information is provided, it must be placed on the front of each sheet and centered within the top margin.

Annotated Drawing Sheets

A marked-up copy of any amended drawing figure, including annotations indicating the changes made, may be submitted or required by the examiner. The annotated drawing sheets must be clearly labeled as "Annotated Marked-up Drawings" and accompany the replacement sheets.

Timing of Corrections

Applicant is required to submit acceptable corrected drawings within the time period set in the Office action. See 37 CFR 1.85(a). Failure to take corrective action within the set period will result in ABANDONMENT of the application.

If corrected drawings are required in a Notice of Allowability (PTOL-37), the new drawings MUST be filed within the THREE MONTH shortened statutory period set for reply in the "Notice of Allowability." Extensions of time may NOT be obtained under the provisions of 37 CFR 1.136 for filing the corrected drawings after the mailing of a Notice of Allowability.

5. Figures 18-21 should be designated by a legend such as --Prior Art-- because only that which is old is illustrated, as acknowledged by Applicant in the Specification, page 4, line 3 through page 6, line 24 and page 19, lines 1-9. See MPEP § 608.02(g). Corrected drawings in compliance with 37 CFR 1.121(d) are required in reply to the Office action to avoid abandonment of the application. The replacement sheet(s) should be labeled "Replacement Sheet" in the page header (as per 37 CFR 1.121(d)) so as not to obstruct any portion of the drawing figures. If the changes are not accepted by the examiner, the applicant will be notified and informed of any required corrective action in the next Office action. The objection to the drawings will not be held in abeyance.

Specification

6. Per Applicant's Request, the title of the invention has been amended.

7. Per Applicant's request, the Specification has been amended. A clean duplicate copy of the Specification has been received. All prior objections to the Specification are hereby withdrawn.

Claim Objections

8. In view of the amendments to claims 5, 20, 12, 16, 30, 32, and 33, prior objections are hereby withdrawn.

9. Claim 15, page 14, line 1, recites "a register allocation stop…", should be –a register allocation step…-- Change 'stop' to 'step'.

10. Claims 26 and 32 are lacking a period at the end of the claim sentence. Add a '.' to the end of each claim.

Claim Rejections - 35 USC § 112

11. In view of the amendments to the claims and the Applicant's comments, the prior 35 USC 112 second paragraph rejections are hereby withdrawn.

Claim Rejections - 35 USC § 103

As Applicant has admitted as Prior Art, in the Background of the Invention, the MUSCAT architecture has a (Specification, page 2, line 11), "plurality of processor elements…integrated on one chip…", (lines 19-22), "Each processor element of the MUSCAT architecture has a FORK instruction for generation of a thread and can create a new thread…" (page 3, lines 20-21), "A countermeasure for such synchronization is called "data dependence assurance". Pages 3 and 4 continue to describe two systems for data dependence assurance: (1) a BLOCK / RELEASE semaphore technique, and (2) a DSP control speculation (cancel/ abandon/ clear execution & restart thread as necessary). Page 7, lines 6-8, "…system should be selected depending upon the situation of data dependence involved…" Page 8, line 2-page 11, line 6 describe Document 4 and (page 8, lines 2-10) "…a mechanism of a translator for converting a machine instruction sequence produced by an ordinary sequential compiler into an instruction sequence for the MUSCAT. A control flow analysis and a data flow analysis are performed for a given machine instruction sequence, and parallelization is attempted using the FORK instruction for each basic block. (Page 8, lines 20-23) "Document 4 further discloses to use profile information to select...basic blocks which exhibits a high branch probability." (page 8, line 24-page 9 line 1), "...data dependence by accessing to a register and a memory between the basic block and a fork destination basic block and basic blocks succeeding the fork destination basic block is investigated." (Page 9, lines 2-12), "...instructions...are reordered...FORK instruction may be positioned on the upstream side as far as possible in the basic block...taking the data dependence relationship into consideration...through a register...instruction is arranged on the upstream side with respect to the FORK...through a memory...the DSPIN instruction or the BLOCK instruction whose argument is a dependent memory address is inserted to the position immediately prior to the FORK instruction.

Applicant disclosed, (page 9, lines 21-25), "...Document 4 further discloses an instruction production procedure unique to the MUSCAT, since the direct relevancy of this to the subject matter of the present invention is poor, the instruction production procedure is not described herein."

Applicant disclosed (page 10, lines 19-25) "...reordering of instructions and processing of a new insertion instruction are different between dependence through a register and dependence through a memory...inside of a compiler (intermediate stage)...it is difficult to perform parallelization processing based on a discrimination between a register and a memory..."

12. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

(a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

13. Claims 1-33 are rejected under 35 U.S.C. 103(a) as being unpatentable over US Patent 6,622,301 to Hirooka et al.,

Per claim 1:

Hirooka disclosed a program conversion technique, which detects branch conversion (loop) opportunities through the use of profiling and statistical analysis. Data dependence is considered. Instructions are reordered to produce parallel code.

A program conversion apparatus for converting a given source program into a program for a mufti-thread processor including a plurality of program counters and a plurality of thread execution apparatus, said plurality of thread execution apparatus being operable to fetch, decode and execute a plurality of instructions of threads simultaneously in accordance with said plurality of program counters such that is possible to execute, after a thread is created, the thread in a control speculative mode wherein a change having had an effect on a register set can be cancelled later, and to execute the thread in a data-dependent speculative mode wherein, when, after a self thread loads a value from a memory location, a parent thread by which the self thread has been created stores a value into the same memory location, at least a processing result of the self thread after the load is abandoned and the processing is re-executed, said mufti-thread processor having an instruction set with which it can be executed by a single machine instruction or a combination of several machine instructions for a thread being executed by any of said thread execution apparatus to create a new thread of the control speculative mode, to end, if a designated condition is satisfied, the self thread and clear the control speculative mode of a thread of the control speculative mode created by the self thread, to abandon the created thread of the control speculative mode, to give, when a thread created by the self thread performs load from a memory location of a designated address, an instruction in advance to temporarily block the operation, to clear the load temporary blocking instruction to the designated memory address, for the thread being executed by the thread execution apparatus to create a new thread of the data-dependent speculative mode and to clear the data-dependent speculative mode of the thread of the data-dependent speculative mode created by the self thread, said program conversion apparatus comprising:

-a register allocation trial section for trying register allocation prior to parallelization to estimate a register allocation situation of variables and intermediate terms of an intermediate program;

Hirooka disclosed (col. 3, lines 9-10) "...profile information, compiler static analysis information..." used to generate a parallel program.

-a fork spot determination section for determining based on a result of the register allocation trial by said register allocation trial section whether or not a conditional branch portion of the intermediate program should be converted into a parallel code for which a thread creation instruction is used and determining a parallelization execution method with the parallel code;

As admitted as prior art by Applicant in the Background of the Invention, (page 8, lines 2-page 10, line 14) MUSCAT architecture has a FORK instruction, and determines a parallelization execution method (BLOCK system or DSP system) when considering data dependence.

-an instruction reordering section for converting the conditional branch portion in the intermediate program into a parallel code for which the thread creation instruction is used based on a result of the determination by said fork spot determination section and referring to the result of the register allocation trial to insert an instruction for assuring a data-dependence relationship between threads through a memory into positions before and after the thread creation instruction and reorder the instructions before and after the thread creation instruction so that thread creation may be performed in an early stage;

As admitted as prior art by Applicant in the Background of the Invention, (page 9, lines 2-12), "…instruction reordering for converting the conditional branch portion in the intermediate program into a parallel code..."

-a register allocation section for performing definite register allocation so that, regarding whether or not a physical register is allocated to the parallelized and reordered instruction sequence, the same allocation result as that upon the register allocating trial may be obtained.

Hirooka disclosed, (Col. 5, lines 39-46) "FIG. 1 is a configuration diagram showing an embodiment of a paralleling compiler…is a function of a paralleling compiler…in which compiler receives as an input a sequential execution source program and produces a parallel program.", col. 3, lines 9-13, "…profile information, compiler static analysis information, or user indication information is obtained to produce a page allocation information to generate a parallel program…" Profiling and analysis are used to determine whether to convert to parallel code.)

Hirooka failed to provide details related to "control speculative mode". Specification, page 4, lines 1-2, "…when it is found that the data is not transferred correctly, it is re-started.
The thread is abandoned, cleared, or otherwise canceled. This technique, as well as the semaphore technique of BLOCK and RELEASE are well known in the art, and admitted as prior art, in the Background of the Invention, as noted above.

Therefore, it would have been obvious, to one of ordinary skill in the art, at the time of the invention, to modify Hirooka's invention which broadly disclosed analysis of intermediate code to improve parallalization, by including details as provided by Applicant in the Background of the Invention, and admitted as prior art, to reject the limitations of claim 1. Profiling, using analysis is useful to produce an optimized parallel program.

Per claim 2:

-fork spot determination section investigates a data dependence relationship through a memory from a basic block in the intermediate program which is a processing object at present to each of basic blocks of branching destinations of a conditional branching instruction positioned at the tail end of the basic block,

-counts, for each of the branching destinations, an instruction step number, from the top of the branching destination basic block, of the instruction at the top one of memory reference instructions in the branching destination basic block which cause the data dependence,

-selects that one of the branching destination basic blocks whose instruction step number is greater as a new thread to be executed parallelly.

(Col. 3, lines 2-5, "...profile information, compiler static analysis information...is obtained to generate...code such that a parallel program is generated..." Generated information is used. Col. 2, lines 53-65, "...parallel program generating method in which loops to be paralleled are detected and then a kernel loop is detected in the loops...code is generated...code is placed before a first execution loop of a main program...to thereby produce a parallel program...This improves data locality...")

As admitted in the Background of the Invention, awareness of the data dependence and basic blocks are used when converting a program to parallel code.

Therefore, it would have been obvious, to one of ordinary skill in the art, at the time of the invention, to have modified Hirooka's invention to include more details regarding control and data dependent speculation, memory accesses and resource (register) optimization, because these details are well known and contribute to efficient compiled parallel code.

Per claim 3:

-fork spot determination section determines the position of a data dependent instruction through a memory in each branching destination basic block using a value

-obtained by accumulating estimated execution cycle numbers of instructions in place, of the instruction step number.

(Col. 3, line 39, "...page most frequently referred to by the processor...", col. 4, lines 8-13, "A profile information version data distribution control method is a method in which for each page, information of a processor which most frequently refers to the page is obtained from profile information ...thereby cause an operating system to optimally distribute data..." Frequency is a metric used to reorder code to parallel form.)

Therefore, it would have been obvious, to one of ordinary skill in the art, at the time of the invention, to have modified Hirooka's invention to include more details regarding control and data dependent speculation, memory accesses and resource (register) optimization, because these details are well known and contribute to efficient compiled parallel code.

Per claims 4:

-upon conversion from a source program into a target program first...

As admitted in the Background of the Invention, (page 8, lines 2-10), "mechanism of a translator for converting a machine instruction sequence produced by an ordinary sequential compiler into an instruction sequence for the MUSCAT(conversion). A control flow analysis and a data flow analysis are performed...parallelization is attempted using the FORK instruction for each basic block..."

-address coordination information for establishing coordination between the basic blocks of the intermediate program in said program conversion apparatus and machine language addresses of the target program to be outputted is outputted together with the target program...

(Hirooka, col. 3, lines 1-7, "...profile information, compiler static analysis information...is obtained to generate...control code such that a parallel program is generated...")

-a target program execution apparatus reads in the target program and the address coordination information and executes the target program and then outputs profile information including branch profile information between basic blocks upon the execution of the target program and data dependence information occurring through a memory between the basic blocks, whereafter...

As admitted in the Background of the Invention, (page 8, lines 2-10), "mechanism of a translator for converting a machine instruction sequence produced by an ordinary sequential compiler into an instruction sequence for the MUSCAT. A control flow analysis and a data flow analysis are performed (profile information)...parallelization is attempted using the FORK instruction for each basic block..."

-when said program conversion apparatus parallelizes the source program to convert the source program into a target program...

As admitted in the Background of the Invention, (page 8, lines 2-10), "mechanism of a translator for converting a machine instruction sequence produced by an ordinary sequential compiler into an instruction sequence for the MUSCAT. A control flow analysis and a data flow analysis are performed ...**parallelization** is attempted using the FORK instruction for each basic block..." (emphasis added)

-said fork spot determination section refers to the profile information to preferentially select a branching destination basic block to which control flows in a high probability at a conditional branch and another branching destination basic block with which data dependence occurs in a low probability at a conditional branch as a new thread to be executed parallelly,

FORK SPOT determination has been admitted as prior art. Hirooka disclosed: (Col. 3, lines 1-6, "...profile information...is obtained to generate a first touch control code such that a parallel program is generated...", col. 3. lines 43-47, "...profile information includes various information obtained by once executing, for example, a parallel program generated...")

Therefore, it would have been obvious, to one of ordinary skill in the art, at the time of the invention, to have modified Hirooka's invention to include more details regarding control and data dependent speculation, memory accesses and resource (register) optimization, because these details are well known and contribute to efficient compiled parallel code.

Per claim 5:

-fork spot determination section produces an instruction to cause a conditional branching destination basic block selected as an execution start point of the new thread to be executed parallelly to temporarily block, when the number of spots of different memory addresses which cause data dependence is smaller than a predetermined number based on a result of an analysis of data dependence through a memory in the intermediate program and a data dependence occurrence probability obtained from the profile information, load operation of the new thread from the memory addresses…

-investigates, when the number of spots of different memory addresses which cause data dependence is equal to or greater than the predetermined number, whether or not the data dependence occurrence probability is lower than a predetermined probability and produces, if the probability is lower, an instruction to create a new thread in the data-dependent speculative mode…

-controls, if the probability is equal to or higher than the predetermined probability, so as to stop the parallelization conversion at the spot.

(Col. 3, lines 2-5, "profile information, compiler static analysis information…is obtained to generate…code such that a parallel program is generated…" Generated information is used to determine whether to create new threads for a parallel program.)

Therefore, it would have been obvious, to one of ordinary skill in the art, at the time of the invention, to have modified Hirooka's invention to include more details regarding control and data dependent speculation, memory accesses and resource (register) optimization, because these details are well known and contribute to efficient compiled parallel code.

Per claim 6:

-fork spot determination section investigates a data dependence relationship through a memory from the basic block in the intermediate program currently which is a processing object at present to each of the branching destination basic blocks of the conditional branching instruction positioned at the tail end of the basic block…

-synthesizes the investigated data dependence relationship and the conditional branching probability obtained from the profile information…

-if a result of the synthesis reveals that the branching probabilities regarding the branching destination basic blocks at the conditional branch do not have a difference greater than a predetermined amount and data dependence occurrence timings through a memory do not have a difference greater than a predetermined amount, said fork spot determination section determines so as not to parallelize the conditional branching portion.

(Col. 3, lines 48-58, "A static analysis information version…compiler generates…static analysis information is analysis information which the compiler can automatically analyze." Profiling and analysis is performed in process to determine whether to convert to parallel code.)

Therefore, it would have been obvious, to one of ordinary skill in the art, at the time of the invention, to have modified Hirooka's invention to include more details regarding control and data dependent speculation, memory accesses and resource (register) optimization, because these details are well known and contribute to efficient compiled parallel code.

Per claim 7:

-syntax analysis section for analyzing the syntax of a source program to produce an intermediate program;

As admitted as prior art, in the Background of the Invention, (page 8, line 2), "…mechanism of a translator for converting…" Also Hirooka disclosed:

(Col. 5, lines 47-54, "In FIG. 1, compiler receives as an input sequential execution source program…produces and outputs parallel program for parallel execution, and generates an intermediate language…")

-parallelization section for performing optimization processing including parallelization for the intermediate program;

As admitted as prior art, in the Background of the Invention, (page 8, line 11), "...Parallelization ...begins with..." Also Hirooka disclosed:

(Col. 5, lines 55-57, "Compiler includes a syntactic analysis section (analyze control flow / data flow) which reads in source program to syntactically analyze program...")

-code generation section for producing a target program including an instruction code for a target processor apparatus from the intermediate program optimized by said parallelization section;

As admitted as prior art, in the Background of the Invention, (page 8, lines 12-13), "...replacement of a branching instruction...with the control speculation mode FORK instruction..."

-said parallelization section including an intermediate program inputting section for reading in the intermediate program and analyzing a control flow and a data flow

As admitted as prior art, in the Background of the Invention, (page 8, line 5-6), "...control flow analysis and a data flow analysis are performed..."

-a register allocation section for trying to perform register allocation prior to parallelization to estimate a register allocation situation of variables and intermediate terms of the intermediate program and executing allocation of registers

As admitted as prior art, in the Background of the Invention, (page 8, line 24-page 9, line 1), "Thereafter, data dependence by accessing to a register and a memory between the basic block and a fork destination basic block and basic blocks succeeding the fork destination basic block is investigated (try register allocation).

Also Hirooka disclosed:

(Col. 6, lines 7-9, "...data allocation information generator section to generate data allocation information of each processor using analysis...")

-a fork spot determination section for determining, based on a result of the trial of the register allocation, a spot of a conditional branch portion of the intermediate program to be converted into a parallel code for which a thread creation instruction is used...

As admitted as prior art, in the Background of the Invention, (page 8, line 7), "...parallelization is attempted using the FORK instruction..." Also Hirooka disclosed:

(Col. 6, lines 23-24, "...detects all loops for execution of the paralleling...")

-an instruction reordering section for performing reordering of instructions before and after the parallelization spot from information of the parallelization spot determined by said fork spot determination section, the data flow...

As admitted as prior art, in the Background of the Invention, (page 8, line 12-13), "...replacement of a branching instruction...with the control speculation mode FORK instruction..." Also, Hirooka disclosed:

(Col. 2, lines 48-60, "...a parallel program generating method in which data is optimally distributed by the kernel loop to thereby improve data locality to increase the processing speed...loops to be paralleled are detected and then a kernel loop is detected in the loops...control code is generated and then the code is placed before a first execution loop of a main program...")

-an intermediate program outputting section for outputting the instruction sequence for which the conversion including the parallelization has been completed in a format of the intermediate program again.
As admitted as prior art, in the Background of the Invention, (page 8, lines 2-5), "Document 4 discloses a mechanism of a translator for converting a machine instruction sequence..." Hirooka also disclosed:
(Col. 5, line 50, "...generates an intermediate language...")

Therefore, it would have been obvious, to one of ordinary skill in the art, at the time of the invention, to have modified Hirooka's invention to include more details regarding control and data dependent speculation, memory accesses and resource (register) optimization, because these details are well known and contribute to efficient compiled parallel code.

Per claim 8:

-parallelization section includes a profile information inputting section for receiving profile information outputted from the target processor apparatus as a result of execution of the target program and converting the profile information into information of an internal format...

As admitted as prior art, in the Background of the Invention, (page 8, lines 5-7), "...control flow analysis and a data flow analysis are performed...and parallelization is attempted..."

-fork spot determination section determines, based on the result of the register allocation trial and the profile information, a spot of a conditional branch portion of the intermediate program to be converted into a parallel code it which the thread generation code is used and determines a parallelization execution method by the parallel code.

As admitted as prior art, in the Background of the Invention, (page 8, lines 5-8), "...control flow analysis and a data flow analysis are performed...and parallelization is attempted...using the FORK instruction..."

Hirooka also disclosed:

(Col. 3, lines 1-6, "...in the parallel program generating method of the present invention, it is also possible that profile information, compiler static analysis information (register allocation information), or user indication information is obtained to generate...code such that a parallel program is generated ...")

Therefore, it would have been obvious, to one of ordinary skill in the art, at the time of the invention, to have modified Hirooka's invention to include more details regarding control

and data dependent speculation, memory accesses and resource (register) optimization, because these details are well known and contribute to efficient compiled parallel code.

Per claim 9:

-instruction reordering section converts conditional branching portions in the intermediate program into a parallel code in which the thread creation instruction is used based on a result of the determination by said fork spot determination section, refers to the result of the register allocation trial to insert an instruction for assuring a data dependence relationship between threads through a memory into positions before and after the thread creation instruction…
As admitted as prior art, in the Background of the Invention, (page 8, line 12-13), "…replacement of a branching instruction…with the control speculation mode FORK instruction…(instruction reordering)"

-reorders the instructions before and after the thread creation instruction so that thread creation may be performed in an early stage.

Hirooka disclosed:

(Col. 2, line 48- col. 3, line 6, "…parallel program generating method…improve data locality…loops to be paralleled are detected…code is generated…to thereby produce a parallel program…profile information, compiler static analysis information…is obtained…")

Therefore, it would have been obvious, to one of ordinary skill in the art, at the time of the invention, to have modified Hirooka's invention to include more details regarding control

and data dependent speculation, memory accesses and resource (register) optimization, because these details are well known and contribute to efficient compiled parallel code.

Per claim 10:

-register allocation section performs definite register allocation so that, regarding whether or not a physical register is allocated to the parallelized and reordered instruction sequence, the same allocation result as that upon the register allocation trial may be obtained.

Hirooka disclosed

(See col. 3, lines 2-3, "profile information, compiler static analysis information...")

Official Notice is given that a reordered instruction sequence should produce the same results when register allocation is decided or the program is not operating correctly. Profile and analysis information should verify this.

Therefore, it would have been obvious, to one of ordinary skill in the art, at the time of the invention, to have modified Hirooka's invention to include more details regarding control and data dependent speculation, memory accesses and resource (register) optimization, because these details are well known and contribute to efficient compiled parallel code.

Per claim 11:

A program conversion apparatus for performing optimization processing including parallelization of an intermediate program obtained by a syntax analysis of a source program performed by a syntax analysis section so that the intermediate program may be suitable for a target processor apparatus, comprising:

-register allocation trial means for trying allocation of registers of the target processor apparatus on the intermediate program and obtaining register allocation information prior to actual allocation;

As admitted as prior art, in the Background of the Invention, (page 8, lines 5-7), "...control flow analysis and data flow analysis are performed...parallelization is attempted..." Consideration is given to registers at page 9, line 7. Also Hirooka disclosed:

(Col. 3, lines 2-3: Static analysis and profiling are done.)

-means for calculating a distance of data dependence generated through a memory in the target processor apparatus for the intermediate program;

(Col. 2, lines 49-50 and 65, "improve data locality...")

-means for determining a fork designation taking the distance of data dependence through a memory on the intermediate program into consideration and replacing a conditional branch with a thread creation instruction; and

An intermediate language is generated (col. 5, line 50) and syntactic analysis is considered.

-means for referring to a result of the register allocation trial to reorder the instructions before and after the thread creation instruction on the intermediate program.

(Col. 2, lines 56- 60, "...code is generated...thereby produce a parallel program. (sequential instructions are reordered to create parallel code)")

Therefore, it would have been obvious, to one of ordinary skill in the art, at the time of the invention, to have modified Hirooka's invention to include more details regarding control and data dependent speculation, memory accesses and resource (register) optimization, because these details are well known and contribute to efficient compiled parallel code.

Per claim 12:

-means for replacing a conditional branch with a thread creation instruction includes:

As admitted as prior art, in the Background of the Invention, (page 8, line 7), "…parallelization (replacing a conditional branch) is attempted using the FORK instruction…"

-means for calculating a minimum value of distance of data dependence of intermediate terms and variables for each of the two branching destinations of the conditional branch;

As admitted as prior art, in the Background of the Invention, (page 8, line 24), "data dependence" is considered.

-means for comparing the two minimum values of the distance of data dependence determined for the two branches of the conditional branch…

As admitted as prior art, in the Background of the Invention, (page 8, lines 5-6 & line 24), "control flow analysis and data flow analysis" …"data dependence" is considered.

-determining, when the two minimum values have a difference greater than or equal to a predetermined value, the branching direction of the branch which exhibits the higher minimum value of the distance of data dependence as a fork destination and selecting the conditional branch spot as a fork spot...

As admitted as prior art, in the Background of the Invention, (page 8, line 7), "...parallelization is attempted using the FORK instruction..." using analysis as noted above.

-determining, when the two minimum values of the distance of data dependence do not have a difference equal to or greater than the predetermined value, that branching destination which has been a branching destination in the original intermediate program as a fork destination...

As noted above, analysis is used when determining a FORK destination.

-selecting the conditional branch spot as a fork spot candidate.

As admitted as prior art, in the Background of the Invention, (page 8, line 7), "...paralleization is attempted using the FORK instruction..."

Also Hirooka disclosed:

(Col. 2, lines 50 and 65, "...improve data locality..." Distance is considered. Static and profiling of code is done.)

Therefore, it would have been obvious, to one of ordinary skill in the art, at the time of the invention, to have modified Hirooka's invention to include more details regarding control

and data dependent speculation, memory accesses and resource (register) optimization, because these details are well known and contribute to efficient compiled parallel code.

Per claim 13:

-means for receiving profile information outputted from a processor apparatus which executes the target program outputted from said program conversion apparatus...

As admitted as prior art, in the Background of the Invention, (page 8, lines 5-6), "control flow analysis and data flow analysis are performed..."

-calculating a conditional branching probability and a data dependence occurrence frequency from the profile information;

Hirooka disclosed: Frequency is considered (col. 3, lines 35-39).

-means for determining a fork destination and a data dependence assurance system from the distance of data dependence, the conditional branch probability and the data dependence occurrence frequency, and the number of spots of different memory addresses which cause data dependence...

As admitted as prior art, in the Background of the Invention, (page 8, lines 2-26), analysis is performed, data dependence is considered and memory addresses which cause data dependence is disclosed.

-replacing the conditional branch with the thread creation instruction.

As admitted as prior art, in the Background of the Invention, (page 8, line 7), "...parallelization is attempted using the FORK instruction..."

Also Hirooka disclosed:

(Col. 3, lines 1-6, Profile and static analysis is used. Frequency is considered (col. 3, lines 35-39). Branching is converted to parallel code (col. 5, lines 60-62), "inserts the code into intermediate language to convert parallel processing thereof by a plurality of processors, and a code generator section which generates and outputs a parallel program using intermediate language converted.")

(Col. 3, lines 1-6, "profile information, compiler static analysis information...is obtained to generate...code such that a parallel program is generated..." Data dependency, branch probabilities, frequency and memory addresses to determine a fork destination are analyzed. Loop instructions (conditional branch) are converted to parallel code (thread creation).)

Therefore, it would have been obvious, to one of ordinary skill in the art, at the time of the invention, to have modified Hirooka's invention to include more details regarding control and data dependent speculation, memory accesses and resource (register) optimization, because these details are well known and contribute to efficient compiled parallel code.

Per claim 14:

-target processor apparatus is a mufti-thread processor which includes a plurality of program counters and a plurality of thread execution apparatus...

-said plurality of thread execution apparatus being operable to fetch, decode and execute a plurality of instructions of threads simultaneously in accordance with said plurality of program counters such that it is possible to execute, after a thread is created, the thread in a control speculative mode …

-wherein a change having had an effect on a register set can be canceled later…

-and to execute the thread in a data-dependent speculative mode wherein, when,

-after a self thread loads a value from a memory location,

-a parent thread by which the self thread has been created stores a value into the same memory location, at least a processing result of the self thread after the load is abandoned

-and the processing is re-executed said mufti-thread processor having an instruction set with which it can be executed by a single machine instruction or a combination of several machine instructions for a thread being executed by any of said thread execution apparatus

- to create a new thread of the control speculative mode,

- to end, if a designated condition satisfied the self thread

- and clear the control speculative mode of a thread of the control speculative mode created by the self thread,

- to abandon the created thread of the control speculative mode,

- to give, when .alpha. thread created by the self thread performs load from a memory location of a designated address,

    - an instruction in advance to temporarily block the operation,
    - to clear the load temporary blocking instruction to the designated memory address, for the thread being executed by the thread execution apparatus to create a new thread of the data-dependent speculative mode and
    - to clear the data-dependent speculative mode of the thread of the data-dependent speculative mode created by the self thread.

See limitations addressed in claim 1 above.

Per claim 15:

A program conversion method for performing an optimization process including parallelization for an intermediate program outputted as a result of a syntax analysis on a program conversion apparatus which compiles a source program and outputs a target program for a target processing apparatus of the mufti-thread type, comprising:

-a register allocating trial step of trying register allocation prior to parallelization to estimate a register allocation situation of variables and intermediate terms of the intermediate program;

-a fork spot determination step of determining based on a result of the register allocation trial whether or not a conditional branch portion of the intermediate program should be converted into a parallel code for which a thread creation instruction is used or performing determination of whether or not the conditional branch portion should be converted into a parallel code and, when such conversion should be performed, determination of a parallelization execution method;

-an instruction reordering step of converting the conditional branch portion in the intermediate program into a parallel code for which the thread creation instruction is used based on a result of the determination by the fork spot determination step and referring to the result of the register allocation trial to insert an instruction for assuring a data-dependence relationship between threads through a memory into positions before and after the thread creation instruction and reorder the instructions before and after the thread creation instruction so that thread creation may be performed in an early stage;

-a register allocation stop of performing definite register allocation so that the same allocation result as that upon the register allocation trial may be obtained for the parallelized and reordered instruction sequence.

(See rejection of limitations as addressed in claim 1 above.)

Per claim 16:

-the fork spot determination step means calculates a minimum value of distance of data dependence of intermediate terms and variables or each of the two branching destinations of the conditional branch, compares the two minimum values of the distance of data dependence determined for the two branches of the conditional branch, determines, when the two minimum values have a difference equal to or greater than a predetermined value, the branching direction of the branch which exhibits the higher minimum value of the distance of data dependence as a fork destination and selects the conditional branch spot as a fork spot, and determines, when the two minimum values of the distance of data dependence do not have a difference equal to or greater than the predetermined value, that branching destination on which has been a branching destination in the original intermediate program as a fork destination and selects the conditional branch spot as a fork spot candidate.

(See rejection of limitations in claim 12 above.)

Per claim 17:

-the fork spot determination step investigates a data dependence relationship through a memory from a basic block having no branch and no confluence from within the intermediate program which is a processing object at present to each of basic blocks of branching destinations of a conditional branching instruction positioned at the tail end of the basic block, counts, for each of the branching destinations, an instruction step number, from the top of the branching destination basic block, of the instruction at the top one of memory reference instructions in the branching destination basic block which cause the data dependence, and selects that one of the branching destination basic blocks whose instruction step number is greater as a new thread to be executed parallelly.

(See rejection of limitations in claim 2 above.)

Per claim 18:

-the fork spot determination step determines the position of a data dependent instruction through a memory in each branching destination basic block using a value obtained by accumulating estimated execution cycle numbers of instructions in place of the instruction step number.

(See rejection of limitations in claim 3 above.)

Per claim 19:

-upon conversion from a source program into a target program first by said program conversion apparatus, address coordination information for establishing coordination between the basic blocks of the intermediate program and machine language addresses of the target program to be outputted is outputted together with the target program, and a processor apparatus which is to execute the object program reads in the target program and the address coordination information and executes the target program and then outputs profile information including branch profile information between basic blocks upon the execution of the target program and data dependence information occurring through a memory between the basic blocks, whereafter, when said program conversion apparatus parallelizes the source program to convert the source program into a target program, the fork spot determination step refers to the profile information to preferentially select a branching destination basic block to which control flows in a high probability at a conditional branch and another branching destination basic block with which data dependence occurs in a low probability at a conditional branch as a new thread to be executed parallelly.

(See rejection of limitations in claim 4 above.)

Per claim 20:

-the fork spot determination step produces an instruction to cause a conditional branching destination basic block selected as an execution start point of the new thread to be executed parallelly to temporarily block, when the number of spots of different memory addresses which cause data dependence is smaller than a predetermined number based on a result of an analysis of data dependence through a memory in the intermediate program and a data dependence occurrence probability obtained from the profile information, load operation of the new thread from the memory addresses, and investigates, when the number of spots of different memory addresses which cause data dependence is equal to or greater than the predetermined number, whether or not the data dependence occurrence probability is lower than a predetermined probability and produces, if the probability is lower, an instruction to create a new thread in the data-dependent speculative mode and controls, if the probability is equal to or higher than the predetermined probability, so as to stop the parallelization conversion at the spot.

(See rejection of limitations in claim 5 above.)

Per claim 21:

-the fork spot determination step investigates a data dependence relationship through a memory from the basic block in the intermediate program currently which is a processing object at present to each of the branching destination basic blocks of the conditional branching instruction positioned at the tail end of the basic block and synthesizes the investigated data dependence relationship and the conditional branching probability obtained from the profile information, and if a result of the synthesis reveals that the branching probabilities regarding the branching destination basic blocks at the conditional branch do not have a difference greater than a predetermined amount and data dependence occurrence timings through a memory do not have a difference greater than a predetermined amount, said fork spot determination section determines so as not to parallelize the conditional branching portion.

(See rejection of limitations in claim 6 above.)

Per claim 22:

-the fork spot determination step includes the steps of:

As admitted as prior art, in the Background of the Invention, (page 8, line 7), "...parallelization is attempted using the FORK instruction...

Also Hirooka disclosed:

(Col. 3, lines 2-4, "...profile information, compiler static analysis information...is obtained...")

-discriminating whether or not the conditional branching instruction corresponds to a return branch of a loop structure in the intermediate program;

As admitted as prior art, in the Background of the Invention, (page 8, line 18-23),

Loop constructs as related to parllelization is disclosed.

Also Hirooka disclosed:

(Col. 3, lines 2-4, "...profile information, compiler static analysis information...is obtained...")

-determining, when the conditional branching instruction corresponds to a loop return branch, the direction of the return branch, which is a loop continuing direction, as a fork destination and selecting the conditional branch spot as a fork spot;

As admitted as prior art, in the Background of the Invention, (page 8, line 18-23),

Loop constructs as related to parllelization is disclosed.

Also Hirooka disclosed:

(Col. 3, lines 2-4, "...profile information, compiler static analysis information...is obtained...")

-calculating, when the conditional branching instruction is not a loop return branch, a minimum value of the distance of data dependence of intermediate terms/variables for each of the two branch destinations of the conditional branch;
As admitted as prior art, in the Background of the Invention, , (page 8, line 18-23),
Loop constructs as related to parllelization was disclosed and (page 8, line 24), "data dependence" was disclosed.
Also Hirooka disclosed:
(Col. 3, lines 2-4, "...profile information, compiler static analysis information...is obtained...")

-comparing the two minimum values of the distance of data dependence determined with regard to the two branches of the conditional branch with each other to discriminate whether or not the two minimum values have a difference greater than a predetermined value;
As admitted as prior art, in the Background of the Invention, (page 8, line 24), "data dependence" was disclosed.
Also Hirooka disclosed:
(Col. 3, lines 2-4, "...profile information, compiler static analysis information...is obtained...")

-determining, when the two minimum values. of the distance of data dependence have a difference greater than the predetermined value, the branch which exhibits a larger one of the minimum values of the distance of data dependence as a fork destination and selecting the conditional branching spot as a fork spot;

As admitted as prior art, in the Background of the Invention, (page 8, line 24), "data dependence" was disclosed in the analysis.

Also Hirooka disclosed:

(Col. 3, lines 2-4, "...profile information, compiler static analysis information...is obtained...")

-determining, when the two minimum values of the distance of data dependence do not have a difference greater than the predetermined value, that one of the branches which has been a branch destination in the original intermediate program as a fork destination and selecting the conditional branching spot as a fork candidate.

As admitted as prior art, in the Background of the Invention, (page 8, line 24), "data dependence" was disclosed in the analysis.

Also Hirooka disclosed:

(Col. 3, lines 2-4, "...profile information, compiler static analysis information...is obtained...")

-if it is determined that a conditional branch is a return branch of a loop, determine direction of return loop, and select branch as a fork spot. (Abstract: lines 6-11, "...a kernel loop having a longest sequential execution time is detected in the source program, Next, a data access pattern equal to that of the kernel loop is reproduced to generate a control code...The first touch control code generated is inserted in the parallel program.", col. 2, lines 54-55, "...loops (return branch) to be paralleled are detected and then a kernel loop is detected in the loops.")

-if it is not a loop, calculate a value related to distance of data dependence for the branches, use calculations to determine fork spot. (Abstract: lines 6-11, "...a kernel loop having a longest sequential execution time is detected in the source program, Next, a data access pattern (value related to distance of data dependence) equal to that of the kernel loop is reproduced to generate a control code...The first touch control code generated is inserted in the parallel program." Profile and static analysis information is obtained (col. 3, lines 2-3).)

Therefore, it would have been obvious, to one of ordinary skill in the art, at the time of the invention, to have modified Hirooka's invention to include more details regarding control and data dependent speculation, memory accesses and resource (register) optimization, because these details are well known and contribute to efficient compiled parallel code.

Per claim 23:

-the distance of data dependence

is the number of steps in the intermediate program which represents at what distance from the top of a basic program of the branching destination the memory reference instruction is positioned

with regard to each of the intermediate terms and variables which are defined in the basic block

which is a processing object at present and may possibly he referred to in the branching destination and besides are estimated to be arranged on the memory.

As admitted as prior art, in the Background of the Invention, (page 8, line 24), "data dependence" was disclosed.

Also Hirooka disclosed:

(Col. 3, lines 2-4, "...profile information, compiler static analysis information...is obtained...")

Therefore, it would have been obvious, to one of ordinary skill in the art, at the time of the invention, to have modified Hirooka's invention to include more details regarding control and data dependent speculation, memory accesses and resource (register) optimization, because these details are well known and contribute to efficient compiled parallel code.

Per claim 24:

-upon the determination of the distance of data dependence, the number of cycles estimated to be required when pertaining instructions are executed on a processor of an object architecture.

(Col. 3, lines 2-4, "...profile information, compiler static analysis information...is obtained...")

Therefore, it would have been obvious, to one of ordinary skill in the art, at the time of the invention, to have modified Hirooka's invention to include more details regarding control and data dependent speculation, memory accesses and resource (register) optimization, because these details are well known and contribute to efficient compiled parallel code.

Per claim 25:

-the instruction reordering step includes:

-a first step of investigating an allocation situation of registers with regard to whether each of intermediate terms and variables in the intermediate program is coordinated with a register or a memory;

-a second step of replacing a branching instruction positioned at the tail end of a basic instruction which is a processing object at present with a control speculation mode FORK instruction while the fork destination which is an operand of the control speculation mode FORK instruction is determined as the fork destination selected by the fork spot determination step;

-a third step of moving a branching condition expression positioned immediately prior to the control speculation mode FORK instruction in the intermediate, instruction to the position immediately next to the control speculation mode FORK instruction and inserting to be tail end of the basic block, which is the destination of the movement of the branching condition expression, an instruction sequence for ending, when the branching condition is satisfied, the self thread and placing a child thread into a settlement mode which is a noncontrol speculation mode, abandoning, when the branching condition is not satisfied, the child thread and keeping the self thread to continue execution of a succeeding instruction train;

-a fourth step of moving each of instruction statements which are on the upstream side with respect to the control speculation mode FORK instruction in the basic block being a processing object at present and are to be substituted into intermediate terms and variables coordinated with a memory to a position on the downstream side with respect to the control speculation FORK instruction and inserting a BLOCK setting instruction to the position immediately prior to the control speculation mode FORK instruction while inserting a BLOCK clear instruction to the position immediately next to the movement destination of the substitute statement; and

-a fifth step of issuing an instruction to allocate the registers in accordance with the register allocation situation assumed by the fork conversion processing in the second step.

(Col. 2, line 47-col. 3, line 6, "...parallel program generating method...improve data locality...loops to be paralleled are detected...code is generated...to thereby produce a parallel program...profile information, compiler static analysis information...is obtained...")

Therefore, it would have been obvious, to one of ordinary skill in the art, at the time of the invention, to have modified Hirooka's invention to include more details regarding control and data dependent speculation, memory accesses and resource (register) optimization, because these details are well known and contribute to efficient compiled parallel code.

Per claim 26:

-the fork spot determination step includes:

-a first step of discriminating whether or not the conditional branching instruction corresponds to a return branch of a loop structure in the intermediate program;

-a second step of provisionally determining, when the conditional branching instruction corresponds to a return branch of a loop structure, the direction of the return branch as a fork destination;

-a third step of calculating, based on the received profile information, a probability with which a taken side of the conditional branching instruction is selected and another probability with which a fall-through side of the conditional branching instruction is selected;

-a fourth step of discriminating whether or not the two calculated probabilities of the branches have a difference greater than a predetermined value;

-a fifth step of provisionally determining, if the difference between the two probabilities of the branches is greater than the predetermined value, the branch which exhibits the higher probability as a fork destination;

-a sixth step of calculating a minimum value of the distance of data dependence for each of the two branching destinations of the conditional branch;

-a seventh step of comparing the two minimum values of the distance of data dependence determined with regard to the two branches of the conditional branch with each other to discriminate whether or not the two minimum values have a difference greater than a predetermined value;

-an eighth step of determining, when the two minimum values of the distance of data dependence have a difference greater than the predetermined value or no data dependence is found that one of the branches which exhibits a higher one of the minimum values of the distance of data dependence as a fork destination;

-a ninth step of calculating a minimum value of the distance of data dependence for the fork destination provisionally determined in the second step or the fifth step and discriminating whether or not the minimum value of the distance of data dependence of the provisionally determined fork destination is equal to or higher than a predetermined value;

-a tenth step of settling, if the minimum value of the distance of data dependence of the provisionally determined fork destination is equal to or higher than a predetermined value or no data dependence through a memory is found, the fork destination provisionally determined in the second step or the fifth step as a formal fork destination;

-an eleventh step of excepting, when it is determined that the minimum value of distance of data dependence is lower than the predetermined value, the basic block from a fork spot;

-a twelfth step of calculating a data dependence occurrence frequency from the received profile information;

-a thirteenth step of discriminating whether or not the data dependence occurrence frequency is equal to or higher than a fixed level, counting, when the data dependence occurrence frequency is lower than the fixed level, the number of intermediate terms/variables on the memory which may possibly cause data dependence from the basic block of the fork source to the basic block of the fork destination in the intermediate program, discriminating whether or not the count value representing a data-dependent spot number is equal to or higher than a fixed level, determining that a fork according the DSP system is used if the data dependent spot number is equal to or higher than the fixed level and determining that a fork according to the BLOCK system is used if the data-dependent spot number is lower than the fixed level, and providing information of the fork to the FORK instruction in the intermediate program; and

-a fourteenth step of counting, when the data dependence occurrence frequency is equal to or higher than the fixed level, the number of data-dependent variables on the memory and determining that a fork according to the BLOCK system is used if the counted number is smaller than a fixed level ([but]] and removing the basic block from a fork candidate if the counted number is equal to or greater than the fixed level

(See rejection of limitation as addressed in claim 22 above.)

Per claim 27:

-the instruction reordering step includes:

-a first step of investigating an allocation situation of registers with regard to whether each of intermediate terms and variables in the intermediate program is coordinated with a register or a memory;

-a second step of replacing a branching instruction positioned at the tail end of a basic instruction which is a processing object at present with a control speculation mode FORK instruction while the fork destination which is an operand of the control speculation made FORK instruction is determined as the fork destination selected by the fork spot determination step;

-a third step of moving a branching condition expression positioned immediately prior to the control speculation FORK instruction in the intermediate instructional to the position immediately next to the control speculation FORK instruction and inserting to the tail end of the basic block, which is the destination of the movement of the branching condition expression, an instruction sequence for ending, when the branching condition is satisfied, the self thread and placing a child thread into a settlement mode which is a noncontrol speculation mode, and abandoning, when the branching condition is not satisfied, the child thread and keeping the self thread to continue execution or a succeeding instruction train;

-a forth fourth step of checking whether a fork data assurance system of the fork spot determined by the fork spot determination step is a_ BLOCK system or a DSP system;

a fifth step of moving, when the fork data assurance system is the BLOCK system, a memory store statement prior to the fork to a position after the fork, inserting necessary BLOCK setting and BLOCK clear instructions, inspecting, upon the movement, a data dependence relationship and moving only those instructions a change of whose instruction execution order does not change a result of arithmetic operation;

a sixth step of modifying, when the fork data assurance system is the DSP system, the FORK instruction produced by replacement in the second step so that a substitute statement into an intermediate term coordinated with a memory is moved to a position next to the FORK instruction to perform the fork in a data-dependent speculation mode; and

a seventh step of issuing an instruction to allocate the registers in accordance with the register allocation situation assumed by the fork conversion process in the second step.

(See rejection of limitations as addressed in claim 25 above.)

Per claim 28:

-A recording medium on which a program for causing a computer to perform an optimization process including parallelization for an intermediate program outputted as a result of a syntax analysis on a compiler which compiles a source program and produces and outputs a target program for a multi-thread processor apparatus is recorded, the optimization process including:

-a register allocation trial process of trying register allocation prior to parallelization to estimate a register allocation situation of variables and intermediate terms of the intermediate program;

-a fork spot determination process of determining based on a result of the register allocation trial whether or not a conditional branch portion of the intermediate program should be converted into a parallel code for which a thread creation instruction is used or performing determination of whether or not the conditional branch portion should be converted into a parallel code and, when such conversion should be performed, determination of a parallelization execution method;

-an instruction reordering process of converting the conditional branch portion in the intermediate program into a parallel code for which the thread creation instruction is used based on a result of the determination 'by the fork spot determination step and referring to the result of the register allocation trial to insert an instruction for assuring a data-dependent relationship between threads through a memory into positions before and after the thread creation instruction and reorder the instructions before and after the thread creation instruction so that thread creation may be performed in an early stage;

-a register allocation process of performing definite register allocation so that the same allocation result as that upon the register allocation trial with regard to whether a physical register is allocated may be obtained for the parallelized and reordered instruction sequence.

(See rejection of limitations as addressed in claim 1 above.)

Per claim 29:

-the fork spot determination process investigates a data dependence relationship through a memory from a basic block in the intermediate program which is a processing object at present to each of basic blocks of branching destinations of a conditional branching instruction positioned at the tail end of the basic block, counts, for each of the branching destinations, an instruction step number from the top of the branching destination basic block, of the instruction at the top one of memory reference instructions in the branching destination basic block which cause the data dependence, and selects that one of the branching destination basic blocks whose instruction step number is greater as a new thread to be executed parallelly.

(See rejection of limitations as addressed in claim 2 above.)

Per claim 30:

-fork spot determination process determines the position of a data-dependent instruction through a memory in each branching destination basic block using a value obtained by accumulating estimated execution cycle numbers of instructions in place of the instruction process number.

(See rejection of limitations as addressed in claim 3 above.)

Per claim 31:

-upon conversion from a source program into a target program first by said compiler, address coordination information for establishing coordination between the basic blocks of the intermediate program and machine language addresses of the target program to be outputted is outputted together with the target program, and a processor apparatus which is to execute the object program reads in the target program and the address coordination information and executes the target program and then outputs profile information including branch profile information between basic blocks upon the execution of the target program and data dependence information occurring through a memory between the basic blocks, whereafter when said compiler parallelizes the source program to convert the source program into a target program, the fork spot determination process refers to the profile information to preferentially select a branching destination basic block to which control flows in a high probability at a conditional branch and another branching destination basic block with which data dependence occurs in a low probability at a conditional branch as a new thread to be executed parallelly.

(See rejection of limitations as addressed in claim 4 above.)

Per claim 32:

-the fork spot determination process produces an instruction to cause a conditional branching destination basic block selected as an execution start point of the new thread to be executed parallelly to temporarily block, when the number of spots of different memory addresses which cause data dependence is smaller than a predetermined number based on a result of an analysis of data dependence through a memory in the intermediate program and a data dependence occurrence probability obtained from the profile information, load operation of the new thread from the memory addresses, and investigates, when the number of spots of different memory addresses which cause data dependence is equal to or greater than the predetermined number, whether or not the data dependence occurrence probability is lower than a predetermined probability and produces, if the probability is lower, an instruction to create a new thread in the data-dependent speculative mode and controls, if the probability is equal to or higher than the predetermined probability, so as to stop the parallelization conversion at the spot

(See rejection of limitations as addressed in claim 5 above.)

Per claim 33:

-the fork spot determination process investigates a data dependence relationship through a memory from the basic block in the intermediate program currently which is a processing object at present to each of the branching destination basic blocks of the conditional branching instruction positioned at the tail end of the basic block and synthesizes the investigated data dependence relationship and the conditional branching probability obtained from the profile information, and if a result of the synthesis reveals that the branching probabilities regarding the branching destination basic blocks at the conditional branch do not have a difference greater than a predetermined amount and data dependence occurrence timings through a memory do not have a difference greater than a predetermined amount, said fork spot determination section determines so as not to parallelize the conditional branching portion.

(See rejection of limitations as addressed in claim 6 above.)

Response to Arguments

14. Applicant has argued, in substance, the following:

(A) As Applicant has pointed out on page 31, 6th paragraph of Amendment received 24 August 2004, "Hirooka did not teach the recited control speculative mode but that Guffens allegedly supplied these features.

Examiner's Response: Examiner has revised the response to the limitations. Guffens is no longer used in the rejection of the limitations. Applicant has disclosed many features in the Background of the Invention including 'control speculative mode' as being prior art.

(B) As Applicant has pointed out on page 32, 2nd paragraph of Amendment, regarding independent claim 15, the cited references do not teach "the recited register allocating step, fork spot determination step, instruction reordering step or register allocation step."

Examiner's Response: Examiner has revised the response to the limitations. Applicant has disclosed many features in the Background of the Invention including 'the recited register allocating step, fork spot determination step, instruction reordering step or register allocation step" as being prior art, in pages 8-10 of the Specification .

(C) As Applicant has pointed out on page 32, 3rd paragraph of Amendment, regarding independent claim 7, the "cited portion of Hirooka, contains no teaching or suggestion of all of the features recited with regard to this feature,"

Examiner's Response: Applicant has disclosed many features in the Background of the Invention. Hirooka broadly disclosed using profile analysis to convert to a parallel program.

(D) As Applicant has pointed out on page 33, 2nd paragraph of Amendment, regarding claim 11, "there is no teaching of the recited means for calculating a distance of data dependence generated through a memory in the target processor apparatus for the intermediate program."

Examiner's Response: Applicant has disclosed many features in the Background of the Invention. Hirooka broadly disclosed using profile analysis to convert to a parallel program and Applicant has admitted in the Background of the Invention that "data dependence" was a consideration.

Without having an English translation of the Documents 1-5, as noted on the IDS form, it is difficult to state that the claimed limitations are not suggested in the prior art and would be obvious when combined with additional prior art.

Conclusion

15. The prior art made of record and not relied upon is considered pertinent to applicant's disclosure.

Any inquiry concerning this communication or earlier communications from the examiner should be directed to Mary Steelman, whose telephone number is (571) 272-3704. The examiner can normally be reached Monday through Thursday, from 7:00 AM to 5:30 PM If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Tuan Q. Dam can be reached at (571) 272-3695. The fax phone number for the organization where this application or proceeding is assigned is 703-872-9306.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

[illegible]

Mary Steelman [illegible]

11/16/2004

WEI Y. ZHEN
PRIMARY EXAMINER